Задача численного решения антагонистической матричной игры

Описание:

1) Написана функция nash_equilibrium(a), которая принимает матрицу выигрыша и возвращает значение игры и оптимальные стратегии первого и второго игроков.

2) Проиллюстрирована работа кода путем решения нескольких игр и визуализации спектров оптимальных стратегий игроков в Jupyter. В частности, приведены игры, в которых:

- спектр оптимальной стратегии состоит из одной точки (т.е. существует равновесие по Нэшу в чистых стратегиях)
- спектр оптимальной стратегии неполон (т.е. некоторые чистые стратегии не используются)
- спектр оптимальной стратегии полон

Для выполнения задачи необходимо установить библиотеки: numpy(обозначается в программае как np) , которая предоставляет нам возможность работы с матрицами и функциями над ними, scipy, которая предоставляет возможность работы с базовыми математическими функциями и алгоритмами, такими как нахождение минимумов и максимумов, а также многими другими, matplotlib(обозначается в программе как plt), которая требуется для построения графиков и дальнейшей визуализации, а также среду разработки jupyter notebook.

Из библиотеки numpy, обозначаемой нами как np мы использовали следующие функции:

1) np.matrix(a) -возвращает матрицу, сделанную из массива a
2) np.transpose (a)– транспонирует матрицу a
3) np.ones(a) – создаёт и заполняет массив длины a единицами
4) np.arange(i,j) - возвращает массив чисел от i до j (j – не включается)

Из библиотеки matplotlib были использованы следующие функции:

1) plt.figure(a) – создает область для построения изображения(графика), если a – уникален, создаётся новая область
2) plt.scatter(a,b,s) - отмечает точки на нашей области(a – массив x-координат точек, b – массив y-координат точек, которые необходимо построить, s – размер точки(опционально))
3) plt.title() – создает заголовок над графиком
4) plt.ylabel(s, fontsize) и plt.xlabel(s, fontsize) - подписывает координатные оси текстом s(fontsize – размер(опционально))
5) plt.grid() – рисует сетку на графике
6) plt.show() – выводит график на область
7) plt.ylim(bottom) – устанавливают ограничения на координаты на графике(устанавливает абсциссу y = bottom)
8) plt.stem(a,b) – создаёт гистограмму (a – массив x-координат точек, b – массив y-координат верхних точек линий гистограммы), функция строит гистограмму

Из библиотеки scipy была использована функция linprog(c, A_ub, B_ub)- поиск оптимального решения задачи ЛП симплекс методом.

Игра-это процесс, цель которого определяется стратегиями и для первого и второго игроков соответственно, где X, Y- множества стратегий, на которых определены функции F(x, y) и G(x, y), которые являются функциями выигрыша для первого и второго игроков соответственно. Цель первого игрока- максимизировать функцию F(x, y), второго -максимизировать функцию G(x, y). Антагонистической игрой называется игра, в которой цели игроков противоположны, то есть F(x, y)=-G(x, y). Такая игра задается множеством Г=<X, Y, F(x, y)>.

Оптимальная стратегия - $x^* \in Arg\max_{x\in X}\min_{y\in Y} F(x,y)$. Седловой точкой называется пара стратегий (x^0, y^0), таких что F(x, y^0)<=F(x^0, y^0)<= F(x^0, y).. Значение функция F(x, y) в седловой точке называется значением игры, а множество $\{x^0, y^0, v = \mathrm{F}(x^0, y^0)\}$- решением игры в чистых стратегиях, то есть чистая стратегия дает полную определенность того, каким образом игрок продолжит игру. Если седловая точка отсутствует, то решения в чистых стратегиях не существует, необходимо искать решение в смешанных стратегиях. Смешанной стратегией первого игрока называется вероятностное распределение p на множестве чистых стратегий X, где $\sum_{i=1}^m p_i = 1$, то есть стратегия i применяется с вероятностью p_i, для второго игрока аналогично. Матричной игрой называется антагонистическая игра, такая что множества стратегий игроков конечны X={1, …, m}, Y={1, …, n}. Стратегия первого игрока- i, второго- j. Вместо функции выигрыша F(x, y) в матричной игре имеем матрицу A=$(a_{ij})_{m*n}$. На пересечении i-ого и j-ого столбца имеем выигрыш для первого игрока, проигрыш для второго.

Алгоритм решения задачи построен на сведении матричной игры, решение которое представимо в виде v=$\max_{p\epsilon P}\min_{1\le j\le n}\sum^m_{i=1} p_i a_{ij}$, к задаче линейного программирования. С помощью замен переменных: v= $\max_{(u,p)\in B} u$, где $B = \{(u,p) | \sum_i{}^m{}_{=1} p_i a_{ij}, j = 1, \dots, n, \sum_i{}^m{}_{=1} p_i = 1, p_i \ge 0, i = 1, \dots, m\}$

и $z = p_i/u$, получаем решение задачи в виде $v = 1/\sum^m_{i=1} z_i{}^0$, где z^0-оптимальное решение задачи линейного программирования:

$\sum^m_{i=1} z_i \to min$, $\sum^m_{i=1} a_{ij} z_i \ge 1, j = 1, \dots, n,\ \ z_i \ge 0, i = 1, \dots, m.$

В дальнейшем будет использоваться функция linprog из scipy.optimize, которая по умолчанию принимает только положительные параметры, поэтому в начале матрица проверяется на неположительность и преобразуется к положительному виде в случае необходимости. Для этого в ней находится максимальный по модулю отрицательный элемент, после чего все значения в матрице увеличиваются на его абсолютную величину. Для решения задачи в программе используется функция linprog из scipy.optimize, которая принимает следующие параметры: коэффициенты функции(в виде массива), которую необходимо минимизировать для первого игрока, максимизировать для второго (для первого игрока-единицы, для второго игрока единицы со знаком минус), матрицу коэффициентов для неравенств (вида $a * p_1 + b * p_2 \le c$, поэтому для первого игрока берется транспонированная матрица, умноженная на -1, для второго-исходная) и одномерный массив, представляющий собой правую часть неравенств (опять же, для первого игрока это единицы, умноженные на -1, для второго- единицы). Функция linprog использует симплекс метод для поиска оптимального решения задачи ЛП, алгоритм которого заключается в следующем:

- Шаг 1. Привести задачу линейного программирования к канонической форме. Для этого перенести свободные члены в правые части (если среди этих свободных членов окажутся отрицательные, то соответствующее уравнение или неравенство умножить на - 1) и в каждое ограничение ввести дополнительные переменные (со знаком "плюс", если в исходном неравенстве знак "меньше или равно", и со знаком "минус", если "больше или равно").

- Шаг 2. Если в полученной системе m уравнений, то m переменных принять за основные, выразить основные переменные через неосновные и найти соответствующее базисное решение. Если найденное базисное решение окажется допустимым, перейти к допустимому базисному решению.
- Шаг 3. Выразить функцию цели через неосновные переменные допустимого базисного решения. Если отыскивается максимум (минимум) линейной формы и в её выражении нет неосновных переменных с отрицательными (положительными) коэффициентами, то критерий оптимальности выполнен и полученное базисное решение является оптимальным - решение окончено. Если при нахождении максимума (минимума) линейной формы в её выражении имеется одна или несколько неосновных переменных с отрицательными (положительными) коэффициентами, перейти к новому базисному решению.
- Шаг 4. Из неосновных переменных, входящих в линейную форму с отрицательными (положительными) коэффициентами, выбирают ту, которой соответствует наибольший (по модулю) коэффициент, и переводят её в основные. Переход к шагу 2.

Далее с помощью linprog().fun получаем значение исследуемой функции, а по формуле $p^0 = vz^0$ находим массив равный оптимальным стратегиям игроков. Анализируем полученные ответы и выводим информацию о значении игры, полноте/неполноте спектра стратегии и существовании решения в чистых стратегиях. Если у нас всего одно ненулевое значение элемента в матрице для первого игрока, то тогда существует решение в чистых стратегиях, если в матрице нет нулевых значений, то спектр оптимальной стратегии полон, в других случаях спектр оптимальной стратегии неполон.

Задание выполняли: Ефарова Д, Михайлов Д., Стрелецкий Н. 312 группа